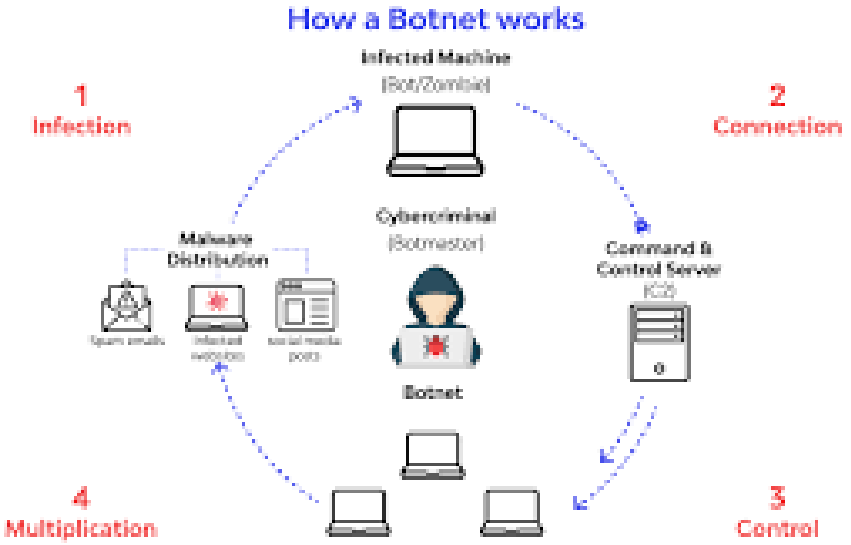
# Building Botnet Detectors using Machine Learning

## Botnet




These are connected computers that perform a number of repetitive tasks to keep websites going. These connected devices make our lives easier and have many applications in devices ranging from computers to electric vehicles to coffee machines.

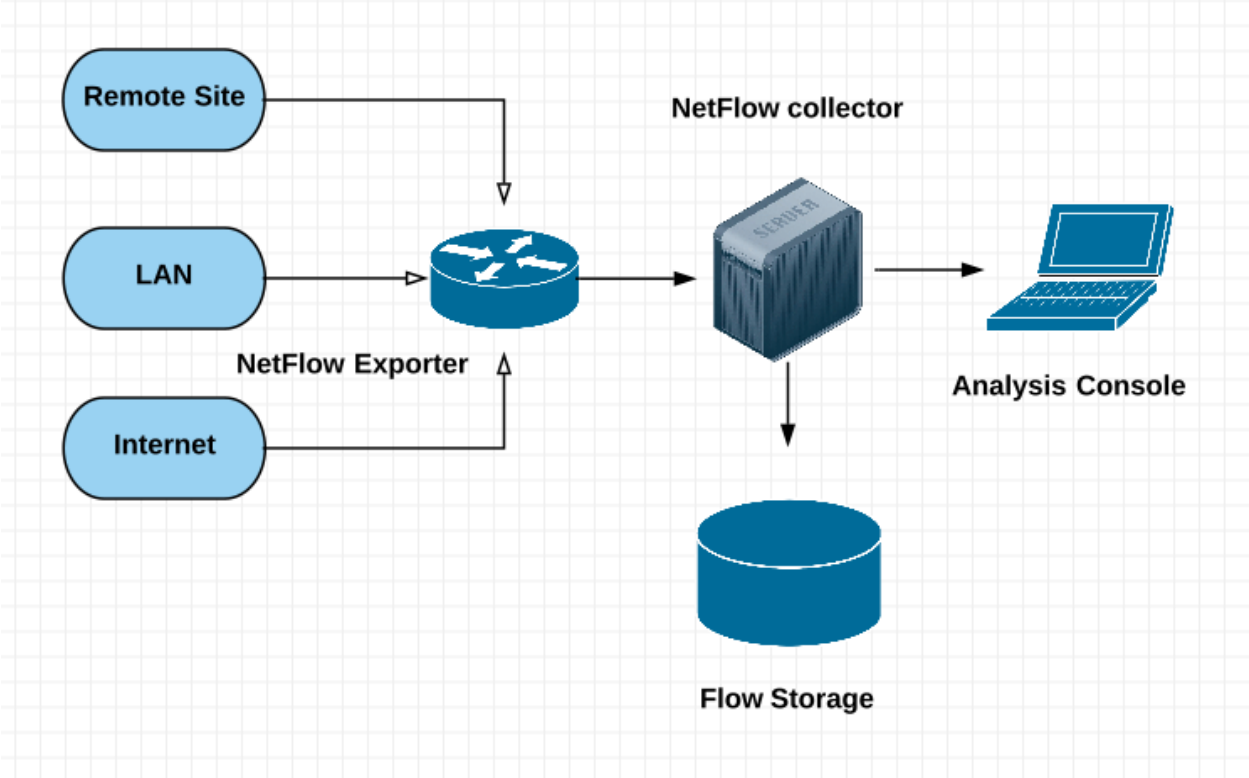
## Vulnerability of Botnets

These devices have a lot of components that can be affected by malware. If one of these components is involved the whole network would get affected. Hence in this project, we are building a novel botnet detection system with python and machine learning techniques.

## The dataset we have used for this project

For this project, we have used the CTU-13 dataset developed by CTU university, Czech Republic. They tried to capture real botnet traffic and background traffic. The dataset is bidirectional NetFlow files. NetFlow is an internet protocol developed by Cisco. The goal of this protocol is to collect IP traffic information and monitor network traffic in order to have a clearer view of the network traffic flow.

## Main Components of NetFlow Architecture




The main components of NetFlow architecture are Netflow exporter, NetFlow Exporter, and Flow Storage. In NetFlow host A sends information to host B and vice versa. This flow is known as bidirectional Netflow.

## Importing the dataset and data preparation

First, we download the tar.bz2 file and then extract the file using Winzip. Then I upload the files into Gdrive and use colab to access them.Then we load the data using the pandas library. The code for the same is shown below:-

```
from google.colab import drive
drive.mount("/content/gdrive")
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
[ ] import pandas as pd
    data = pd.read_csv('/content/gdrive/My Drive/CTU-13/capture20110819.binetflow')
```

Then I call the script DataPreparation.py  which contains a class Prepare which is used select and test data. The second script called LoadData.py which is used to load data from .binetflow files and generate a pickle file.

```
class Prepare(threading.Thread):

    def __init__(self, X, Y, XT, YT, accLabel=None):
        threading.Thread.__init__(self)
        self.X = X
        self.Y = Y
        self.XT=XT
        self.YT=YT
        self.accLabel= accLabel

    def run(self):
        X = np.zeros(self.X.shape)
        Y = np.zeros(self.Y.shape)
        XT = np.zeros(self.XT.shape)
        YT = np.zeros(self.YT.shape)
        np.copyto(X, self.X)
        np.copyto(Y, self.Y)
        np.copyto(XT, self.XT)
        np.copyto(YT, self.YT)
        for i in range(9):
            X[:, i] = (X[:, i] - X[:, i].mean()) / (X[:, i].std())
        for i in range(9):
            XT[:, i] = (XT[:, i] - XT[:, i].mean()) / (XT[:, i].std())
```

```
def run(self):
  X = np.zeros(self.X.shape)
  Y = np.zeros(self.Y.shape)
  XT = np.zeros(self.XT.shape)
  YT = np.zeros(self.YT.shape)
  np.copyto(X, self.X)
  np.copyto(Y, self.Y)
  np.copyto(XT, self.XT)
  np.copyto(YT, self.YT)
  for i in range(9):
    X[:, i] = (X[:, i] - X[:, i].mean()) / (X[:, i].std())
  for i in range(9):
    XT[:, i] = (XT[:, i] - XT[:, i].mean()) / (XT[:, i].std())
```

```
import socket, struct, sys
import numpy as np
import pickle

def loaddata(fileName):


    file = open(fileName, 'r')

    xdata = []
    ydata = []
    xdataT = []
    ydataT = []
    flag=0
    count1=0
    count2=0
    count3=0
    count4=0

    #dicts to convert protocols and state to integers
    protoDict = {'arp': 5, 'unas': 13, 'udp': 1, 'rtcp': 7, 'pim': 3, 'udt': 11, 'esp': 12, 'tcp' : 0, 'rarp': 14, 'ipv6-icmp': 9, 'rtp': 2, 'ipv6': 
    stateDict = {'': 1, 'FSR_SA': 30, '_FSA': 296, 'FSRPA_FSA': 77, 'SPA_SA': 31, 'FSA_SRA': 1181, 'FPA_R': 46, 'SPAC_SPA': 37, 'FPAC_FPA': 2, '_R': 
```

```
 file.readline()

 for line in file:
     sd = line[:-1].split(',')
     dur, proto, Sport, Dport, Sip, Dip, totP, totB, label, state = sd[1], sd[2], sd[4], sd[7], sd[3], sd[6], sd[-4], sd[-3], sd[-1], sd[8]
     try:
         Sip = socket.inet_aton(Sip)
         Sip = struct.unpack("!L", Sip)[0]
     except:
         continue
     try:
         Dip = socket.inet_aton(Dip)
         Dip = struct.unpack("!L", Dip)[0]
     except:
         continue
     if Sport=='': continue
     if Dport=='': continue
     #back, nor, bot
     try:

         if "Background" in label:
             label=0
```

```
    elif "Botnet" in label:
        label = 1


    if flag==0:
        #Training Dataset
        if label==0 and count1<20001:
            xdata.append([float(dur), protoDict[proto], int(Sport), int(Dport), Sip, Dip, int(totP), int(totB), stateDict[state]])
            ydata.append(label)
            count1+=1

        elif label==1 and count2<20001:
            xdata.append([float(dur), protoDict[proto], int(Sport), int(Dport), Sip, Dip, int(totP), int(totB), stateDict[state]])
            ydata.append(label)
            count2+=1

        elif count1>19999 and count2>19999:
            #print("HI")
            flag=1

    else:
        #Test dataset
        if label==0 and count3<5001:
            #print("H")
            xdataT.append([float(dur), protoDict[proto], int(Sport), int(Dport), Sip, Dip, int(totP), int(totB), stateDict[state]])
            ydataT.append(label)
```

```
                    ydataT.append(label)
                    count3+=1
                elif label==1 and count4<5001:
                    xdataT.append([float(dur), protoDict[proto], int(Sport), int(Dport), Sip, Dip, int(totP), int(totB), stateDict[state]])
                    ydataT.append(label)
                    count4 += 1
                elif count3>4999 and count4>4999:
                    break
        except:
            continue

    #pickle the dataset for fast loading
    file = open('flowdata.pickle', 'wb')
    pickle.dump([np.array(xdata), np.array(ydata), np.array(xdataT), np.array(ydataT)], file)

    #return the training and the test dataset
    return np.array(xdata), np.array(ydata), np.array(xdataT), np.array(ydataT)

if __name__ == "__main__":
    loaddata('/content/gdrive/My Drive/CTU-13/capture20110819.binetflow')
```

## Importing the data from the pickle file and using a machine learning classifier

```
import pickle
file = open('flowdata.pickle', 'rb')
data  = pickle.load(file)

Xdata = data[0]
Ydata =  data[1]
XdataT = data[2]
YdataT = data[3]
```

The pickle file is loaded into the data variable and the training and testing data are chosen.

```
from sklearn.tree import *
```

Then we import the tree classifier from sklearn.

```
Prepare(Xdata,Ydata,XdataT,YdataT)
```

Then we prepare the training and testing data.

```
from sklearn import tree
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(Xdata,Ydata,test_size=0.3,random_state=0)

depth = []
for i in range(3,20):
    clf = tree.DecisionTreeClassifier(max_depth=i)
    clf = clf.fit(x_train,y_train)
    depth.append((i,clf.score(x_test,y_test)))
print(depth)
```

Finally, we fit the decision tree classifier to the data and predict accuracy for each height of the decision tree from 3-20.

## References

1.Mastering-Machine-Learning-for-Penetration-Testing/Chapter05 at master · PacktPublishing/Mastering-Machine-Learning-for-Penetration-Testing · GitHub

2.Build botnet detectors using machine learning algorithms in Python [Tutorial] | Packt Hub (packtpub.com)